IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re application of: Steven R. Chalmer et al.			
Appln. No.:	09/605,812	**Art Unit:**	2195
Filed:	June 28, 2000	**Examiner:**	Nilesh R. SHAH
For:	**REPLACEABLE SCHEDULING ALGORITHM IN MULTITASKING KERNEL**	**Docket No.:**	EMS-00801

Certificate of Mailing

I hereby certify that the foregoing documents are being deposited with the United States Postal Service as first class mail, postage prepaid, in an envelope addressed to: Mail Stop Appeal Brief-Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on this date of September 13, 2005.

Bonny Rogers

Name: Bonny Rogers

TRANSMITTAL OF APPEAL BRIEF

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

Applicant hereby submits the originally-signed Appeal Brief with Certificate of Mailing (in triplicate), check in the amount of $500.00, and postcard receipt for the above-referenced patent application.

Although we believe that we have appropriately provided for any fees due in connection with this submission, the Commissioner is authorized to credit any overpayment or charge any deficiencies to/from our **Deposit Account No. 503596**. Two originally-executed copies of this form are being submitted.

Should there be any questions after reviewing this paper, the Examiner is invited to contact the undersigned at 508-898-8603.

Respectfully submitted,
MUIRHEAD AND SATURNELLI, LLC

September 13, 2005
Date

Donald W. Muirhead
Reg. No. 33,978

Muirhead and Saturnelli, LLC
200 Friberg Parkway, Suite 1001
Westborough, MA 01581

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

CERTIFICATE OF MAILING

I hereby certify that the foregoing document is being deposited with the United States Postal Service as first class mail, postage prepaid, "Post Office to Addressee", in an envelope addressed to: Mail Stop Appeal Brief-Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, VA, 22313-1450 on September 13, 2005.

Bonny J. Rogers

* * *

APPEAL BRIEF UNDER 37 C.F.R. § 41.37

Application Serial No.: 09/605,812
Filed: June 28, 2000

Appellants/Applicants: Chalmer, et al.

Title: REPLACEABLE SCHEDULING ALGORITHM
IN MULTITASKING KERNAL

Appeal from a decision of the Primary Examiner
dated March 17, 2005

Atty. Docket: EMS-00801

09/16/2005 EFLORES 00000029 09605812
01 FC:1402 500.00 OP

REAL PARTY IN INTEREST

The above-identified application is assigned to EMC Corporation by virtue of an Assignment recorded by the U.S. Patent and Trademark Office on June 28, 2000, at Reel 010938, Frame 0345.

RELATED APPEALS AND INTERFERENCES

Appellants are not aware of any appeals or interferences related to the above identified application.

STATUS OF CLAIMS

This is an appeal from a decision of the Primary Examiner in the Final Office Action dated March 17, 2005, finally rejecting Claims 1-3, 5-14, 16-20, 22-31, 33 and 34 in the above-identified patent application. Claims 9 and 26 stand rejected under 35 U.S.C. 102(b). Claims 1-3, 5-8, 10-14, 16-20, 22-25, 27-31, 33 and 34 stand rejected under 35 U.S.C. 103(a). Claims 4, 15, 21, and 32 have been previously cancelled. No claim has been allowed. A Notice of Appeal was submitted on July 14, 2005.

STATUS OF AMENDMENTS

Appellants filed an after-final Response on April 22, 2005, in which no claim amendments were made. Therefore, all amendments proposed by the Appellants throughout previous prosecution of this case have been entered. The claims involved in this Appeal are set forth in the attached Claims Appendix.

SUMMARY OF CLAIMED SUBJECT MATTER

Prior to discussing the invention specifically, it may be useful to briefly provide general information to aid in the understanding of the invention.

A scheduler is a special type of operating system process that manages the running of other processes on a processor. The other processes may be user applications, other operating system processes, or any other processes that may be scheduled by the scheduler. In the simplest case, a single scheduler schedules a plurality of processes to run on a single processor. The scheduler may use a particular algorithm to determine which processes run, in what order, and for how long. For example, the scheduler may use a round robin technique where each process that is available to run is scheduled one at a time and, after each process has been scheduled once, each process is then rescheduled, etc. There are other scheduling algorithms that use different algorithms to determine which process runs, in what order, and for how long.

The present claimed invention is for a system that has a plurality of available schedulers, a particular one of which may be selected for use at any one time. Note that this may be contrasted from a plurality of processes that are scheduled by the scheduler and the claims specifically recite that the scheduler is different from the processes being scheduled by the scheduler. The particular scheduler that is chosen for use depends upon run time conditions.

Turning specifically to the claims, claim 1 recites a method of providing a particular scheduler for a multitasking system for a processor. The method includes choosing the

particular scheduler from a plurality of schedulers, where at least one of the plurality of schedulers selects processes to be run from a plurality of runable processes different from the plurality of schedulers and where choosing a particular scheduler is based on parameters that vary according to run time conditions, setting a program counter to an address corresponding to code of the particular scheduler, and the processor executing code at an address corresponding to the program counter. Claims 2, 3, and 5-8 depend directly or indirectly from claim 1.

Claim 9 recites a method of scheduling tasks in a multitasking operating system. The method includes choosing a particular scheduler from a plurality of schedulers, where at least one of the plurality of schedulers selects processes to be run from a plurality of runable processes different from the plurality of schedulers and where choosing a particular scheduler is based on parameters that vary according to run time conditions, and running the particular scheduler to schedule tasks. Claims 10-14, 16, and 17 depend directly or indirectly from Claim 9.

Claim 18 recites computer software in combination with a computer readable medium that provides a particular scheduler for a multitasking system for a processor. The software includes executable code, provided on a computer readable medium, that chooses the particular scheduler from a plurality of schedulers, where at least one of the plurality of schedulers selects processes to be run from a plurality of runable processes different from the plurality of schedulers and where executable code chooses a particular scheduler using parameters that vary according to run time conditions, executable code, provided on a computer readable medium, that sets a program counter to an address corresponding to code

of the particular scheduler; and executable code, provided on a computer readable medium, that causes the processor to execute code at an address corresponding to the program counter. Claims 19, 20, and 22-24 depend directly or indirectly from claim 18.

Claim 26 recites computer software in combination with a computer readable medium that schedules tasks in a multitasking operating system. The software includes executable code, provided on a computer readable medium, that chooses the particular scheduler from a plurality of schedulers, where at least one of the plurality of schedulers selects processes to be run from a plurality of runable processes different from the plurality of schedulers and where choosing a particular scheduler is based on parameters that vary according to run time conditions, and executable code, provided on a computer readable medium, that runs the particular scheduler to schedule tasks. Claims 27-31, 33, and 34 depend directly or indirectly from Claim 26.

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

I. Claims 9 and 26 stand rejected under 35 U.S.C. 102(b) as being anticipated by U.S. Patent No. 6,108,683 to Kamada et al. (hereinafter "Kamada").

II. Claims 1-3, 5-8, 10-14, 16-20, 22-25, 27-31, 33 and 34 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Kamada and further in view of U.S. Patent No. 5,630,130 to Perotto, et al. (hereinafter "Perotto").

ARGUMENT

I. The Examiner has failed to establish a prima-facie case of anticipation under 35 U.S.C. §102(b) of Claims 9 and 26 over Kamada.

A. Anticipation Standard.

To establish a proper case of anticipation in rejecting claims under 35 U.S.C. §102(b), it is necessary for the Examiner to demonstrate that each element of the claim in issue is found, either expressly described or under principles of inherency, in a single prior art reference, or that the claimed invention was previously known or embodied in a single prior art device or practice. See Kalman v. Kimberly-Clark Corporation, 713 F.2d 760, 771 , 218 USPQ 781, 789 (Fed. Cir. 1983). The exclusion of a claimed element from a prior art reference is enough to negate anticipation by that reference. Atlas Powder Co. v. E.I. Du Pont De Nemours & Co., 750 F.2d 1569, 1574, 224 USPQ 409, 411 (Fed. Cir. 1984).

B. The cited reference does not teach every element of Appellants' claimed invention.

Appellant s' independent claims 9 and 26 recite some form of choosing a particular scheduler from among a plurality of schedulers according to run time conditions and specifically recite that the schedulers are different from the processes that are scheduled by the scheduler. As explained in more detail below, although Kamada may arguably disclose multiple schedulers, there is no teaching or suggestion in Kamada regarding choosing a particular scheduler or, for that matter, choosing a particular scheduler based on run time conditions.

Kamada discloses a computer-system fixed-priority process scheduler that is supported by an operating system and that establishes fixed priorities respectively corresponding to a plurality of processes to be scheduled. Kamada discloses embodiments in Figure 2 and Figure 5 which indicate the possibility that there may be more than one scheduler. In the case of Figure 2, Kamada teaches that the user-level process scheduler operating at the priority of 159 is divided into two schedulers, namely, the user-level process schedulers 10-1 and 10-2. For Figure 5, Kamada also indicates that the user-level process scheduler operating at the priority of 159 is divided into two schedulers: the user-level process schedulers 10-1 and 10-2.

Column 7, lines 1-40 of Kamada disclose changing the priority of processes scheduled by the scheduler and how to swap tasks scheduled by a scheduler. There is no mention in column 7, lines 1-40 of multiple schedulers and, in fact, all instances mentioning the scheduler in column 7, lines 1-40, are singular, thus indicating that only one scheduler is being discussed.

Column 11, lines 21-37 mention multiple schedulers and state:

> FIG. 5 illustrates yet another example of (the operating environment of) a user-level process scheduler of the present invention. A characteristic aspect of this example resides in that there are provided a plurality of systems (namely, a plurality of schedulers, each of which is equivalent to the example of the user-level process scheduler of FIG. 1) correspondingly to the real time class. Namely, there are provided two user-level process schedulers 10-1 and 10-2 which operate at the priority of 159 of the real time class. Moreover, a group of the user processes 12-1 and 12-2 and another group of the user processes 12-3 and 12-4 (incidentally, the user processes 12-1 to 12-4 operate at the priority of 158) are respectively provided as user process groups 11-1 and 11-2 which are objects of the scheduling. In the case of this example, the user-level process scheduler 10-1 performs the scheduling of the user processes 12-1 and 12-2. Further,

the user-level process scheduler 10-2 performs the scheduling of the user processes 12-3 and 12-4.

Lines 37-40 go on to state:

Needless to say, FIG. 5 illustrates the case that there are two user-level process schedulers 10-1 and 10-2 in the operating system. However, if necessary, there can be provided user-level process schedulers of an arbitrary number.

There is no discussion in Kamada relating to choosing between the schedulers 10-1, 10-2 or using any criteria whatsoever (including run time conditions) to choose between the schedulers 10-1, 10-2. Rather, Kamada merely discloses that there can be more than one scheduler, each of which schedules its own set of user processes (e.g., the scheduler 10-1 schedulers user processes 12-1 and 12-2 while the scheduler 10-2 schedules user processes 12-3 and 12-4).

In addition, all of the figures of Kamada that illustrate more than one scheduler appear to show the multiple schedulers operating concurrently in memory. See, for example, figure 2 and figure 5. This is consistent with the description in Kamada, which does not appear to distinguish between the schedulers or discuss how to choose or prioritize between the schedulers. Said differently, there is no choice to be made between the schedulers of Kamada because the plurality of schedulers appear to be operating concurrently. In contrast, Appellants' claims 9 and 26 specifically recite choosing a particular scheduler from a plurality of schedulers, which of course is inconsistent with the plurality of schedulers operating concurrently as taught by Kamada.

Thus, the features recited in Appellants' claims 9 and 26 of choosing a particular scheduler from a plurality of schedulers according to run time conditions where the particular

scheduler is different from the plurality of schedulers is simply not found in the Kamada reference. In fact, Kamada does not appear to include any teaching or suggestion of any mechanism for choosing a particular one of the plurality of schedulers he discloses let alone what criteria would be used for any sort of selection. To the extent Kamada discloses multiple schedulers, Kamada only shows the schedulers as running at the same time in parallel and thus there is no choosing from among a plurality of schedulers as recited in Appellants' claims 9 and 26.

Accordingly, for the reasons noted above, it is requested that the Board reverse the Examiner's rejection under 35 U.S.C. 102(b).

II. The Examiner has failed to establish a prima-facie case of obviousness under 35 U.S.C. §103(a) of Claims 1-3, 5-8, 10-14, 16-20, 22-25, 27-31, 33 and 34 as being unpatentable over Kamada and further in view of Perotto.

A. Obviousness Standard.

In determining whether or not there is a proper case of obviousness, it is necessary to establish whether one of ordinary skill in the art would, having the references before him, be motivated to make the proposed combination, modification or substitution so as to yield all elements of a claimed invention. In re Lintner, 458 F.2d 1013, 1016 (CCPA, 1972). The Board has previously laid out the legal analysis for determining obviousness and the corresponding burdens faced by the Examiner and the Appellant as set forth below.

In rejecting claims under 35 U.S.C. §103, it is incumbent upon the Examiner to establish a factual basis to support the legal conclusion of obviousness. See In re Fine, 837

F.2d 1071, 1073, 5 USPQ2d 1596, 1598 (Fed. Cir. 1988). In so doing, the Examiner is expected to make the factual determinations set forth in Graham v. John Deere Co., 383 U.S. 1, 17, 148 USPQ 459, 467 (1966), and to provide a reason why one having ordinary skill in the pertinent art would have been led to modify the prior art or to combine prior art references to arrive at the claimed invention. Such reason must stem from some teaching, suggestion or implication in the prior art as a whole or knowledge generally available to one having ordinary skill in the art. Uniroyal, Inc. v. Rudkin-Wiley Corp., 837 F.2d 1044, 1051, 5 USPQ2d 1434, 1438 (Fed. Cir. 1988); Ashland Oil, Inc. v. Delta Resins & Refractories, Inc., 776 F.2d 281, 293, 227 USPQ 657, 664 (Fed. Cir. 1985); ACS Hosp. Sys., Inc. v. Montefiore Hosp., 732 F.2d 1572, 1577, 221 USPQ 929, 933 (Fed. Cir. 1984). Further, prior art references should not be combined where such combinations would render inoperable the intended purposes disclosed therein. In re Gordon, 733 F.2d 900, 902, 221 USPQ 1125, 1127 (Fed. Cir. 1984). These showings by the Examiner are an essential part of complying with the burden presenting a prima facie case of obviousness. Note In re Oetiker, 977 F.2d 1443, 1445, 24 USPQ2d 1443, 1444 (Fed. Cir. 1992). If that burden is met, the burden then shifts to the Appellant to overcome the prima facie case with argument and/or evidence. Obviousness is then determined on the basis of the evidence as a whole. See id.; In re Hedges, 783 F.2d 1038, 1039, 228 USPQ 685, 686 (Fed. Cir. 1986); In re Piasecki, 745 F.2d 1468, 1472, 223 USPQ 785, 788 (Fed. Cir. 1984); and In re Rinehart, 531 F.2d 1048, 1052, 189 USPQ 143, 147 (CCPA 1976).

B. The cited reference does not teach or fairly suggest every element of Appellants' claimed invention as to have rendered Appellants' Claims 1-3, 5-8, 10-14, 16-20, 22-25, 27-31, 33 and 34 obvious to one of ordinary skill in the art at the time the invention was made.

Kamada discloses a computer-system fixed-priority process scheduler that is supported by an operating system and that establishes fixed priorities respectively corresponding to a plurality of processes to be scheduled. Kamada discloses embodiments in figure 2 and figure 5 which indicate the possibility that there may be more than one scheduler. In the case of Figure 2, Kamada teaches that the user-level process scheduler operating at the priority of 159 is divided into two schedulers, namely, the user-level process schedulers 10-1 and 10-2. For Figure 5, Kamada also indicates that the user-level process scheduler operating at the priority of 159 is divided into two schedulers, namely, the user-level process schedulers 10-1 and 10-2.

All of the figures of Kamada that illustrate more than one scheduler appear to show the schedulers operating concurrently in memory. See, for example, figure 2 and figure 5. This is consistent with the description in Kamada, which does not appear to distinguish between the schedulers or discuss how to choose or prioritize between the schedulers. Said differently, there is no choice to be made between the schedulers of Kamada because the plurality of schedulers appear to be operating concurrently.

Perotto discloses a multitasking controller having task storage means (2) for storing up to N tasks (P0,P1,P2,P3) where each task comprises a sequence of instructions.. The controller also includes a microprocessor for processing, by time-sharing, a plurality of such

N tasks, and a random access memory (12) for storing variable data created and used by said microprocessor. The microprocessor further includes a scheduler (7) realized in hardware for controlling the use of the microprocessor or by such processes, and program counter storage means for storing N program counters (Pc0, Pc1, Pc2, Pc3) each for use by the scheduler (7), which is able select a different one of the program counters (Pc0, Pc1, Pc2, Pc3) when the task processed by the microprocessor is changed without the transfer of data from the random access memory (12).

All of the present independent claims recite choosing a particular scheduler from a plurality of schedulers based on parameters that vary according to run time conditions. As discussed above with respect to the rejection of claims 9 and 26 under 35 U.S.C. 102(b), Kamada, by itself, does not show, teach, or suggest this feature because, even if Kamada discloses multiple schedulers, there does not appear to be any disclosure in Kamada relating to choosing any *particular* scheduler or how this would be done and what criteria would be used.

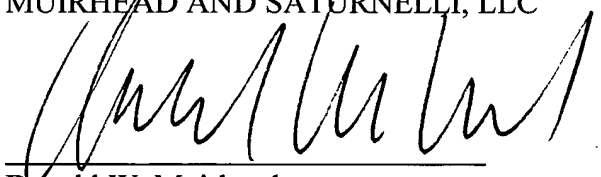
In addition, the deficiencies of Kamada are not overcome by the addition of Perotto, especially since Perotto discloses a *single* scheduler that schedules one of the four disclosed tasks (P0, P1, P2, P3). None of the tasks P0, P1, P2, or P3 of Perotto are themselves schedulers and all of Applicants' independent claims specifically recite some form of the schedulers being different from the tasks scheduled by the schedulers. Since Perotto only discloses a single scheduler, Perotto (like Kamada) can not show, teach, or suggest a recited feature of the present claimed invention relating to choosing a particular scheduler from a plurality of schedulers, where the choice is based on parameters that vary according to run time conditions.

Accordingly, for the reasons noted above, it is requested that the Board reverse the Examiner's rejection under 35 U.S.C. 103(a).

CONCLUSION

In view of the above, it is respectfully requested that the Board reverse all of the Examiner's rejections under 35 U.S.C. 102(b) and 103(a).

Respectfully submitted,
MUIRHEAD AND SATURNELLI, LLC



Date: September 13, 2005

Donald W. Muirhead
Registration No. 33,978

Muirhead and Saturnelli, LLC
200 Friberg Parkway, Suite 1001
Westborough, MA 01581
T: (508) 898-8601
F: (508) 898-8602

CLAIMS APPENDIX

The claims on Appeal are as follows:

1. (Previously Presented) A method of providing a particular scheduler for a multitasking system for a processor, comprising:

choosing the particular scheduler from a plurality of schedulers, wherein at least one of the plurality of schedulers selects processes to be run from a plurality of runable processes different from the plurality of schedulers and wherein choosing the particular scheduler is based on parameters that vary according to run time conditions;

setting a program counter to an address corresponding to code of the particular scheduler; and

the processor executing code at an address corresponding to the program counter.

2. (Previously Presented) A method, according to claim 1, further comprising:

setting a stack pointer to an address corresponding to stack space for the particular scheduler; and

the processor using the stack space at the stack pointer after executing code at the address corresponding to the program counter.

3. (Original) A method, according to claim 1, wherein all of the schedulers use the same stack.

4. (Cancelled)

5. (Previously Presented) A method, according to claim 1, wherein at least one of the schedulers is for statistical code profiling.

6. (Previously Presented) A method, according to claim 1, wherein a first one of the schedulers is for start up conditions and a second one of the schedulers is for steady state operation.

7. (Previously Presented) A method, according to claim 1, wherein choosing the particular scheduler is performed by setting up a return from an exception that causes the scheduler to execute.

8. (Previously Presented) A method, according to claim 1, wherein setting a program counter includes modifying a variable that is modified according to the particular scheduler that is chosen.

9. (Previously Presented) A method of scheduling tasks in a multitasking operating system, comprising:

choosing a particular scheduler from a plurality of schedulers, wherein at least one of the plurality of schedulers selects processes to be run from a plurality of runable processes different from the plurality of schedulers and wherein choosing a particular scheduler is based on parameters that vary according to run time conditions; and

running the particular scheduler to schedule tasks.

10. (Previously Presented) A method, according to claim 9, wherein choosing a particular scheduler is performed by setting up a return from an exception that causes the scheduler to execute.

11. (Previously Presented) A method, according to claim 9, wherein running the particular scheduler includes setting a program counter to an address corresponding to code of the particular scheduler.

12. (Previously Presented) A method, according to claim 11, wherein setting a program counter includes modifying a variable that is modified according to the particular scheduler that is chosen.

13. (Previously Presented) A method, according to claim 9, further comprising:

setting a stack pointer to an address corresponding to stack space for the particular scheduler; and

the processor using the stack space at the stack pointer after executing code at the address corresponding to the program counter.

14. (Original) A method, according to claim 9, wherein all of the schedulers use the same stack.

15. (Cancelled)

16. (Previously Presented) A method, according to claim 9, wherein at least one of the schedulers is for statistical code profiling.

17. (Previously Presented) A method, according to claim 9, wherein a first one of the schedulers is for start up conditions and a second one of the schedulers is for steady state operation.

18. (Previously Presented) Computer software in combination with a computer readable medium that provides a particular scheduler for a multitasking system for a processor, comprising:

executable code, provided on a computer readable medium, that chooses the particular scheduler from a plurality of schedulers, wherein at least one of the plurality of schedulers selects processes to be run from a plurality of runable processes different from the plurality of schedulers and wherein executable code that chooses the particular scheduler uses parameters that vary according to run time conditions;

executable code, provided on a computer readable medium, that sets a program counter to an address corresponding to code of the particular scheduler; and

executable code, provided on a computer readable medium, that causes the processor to execute code at an address corresponding to the program counter.

19. (Previously Presented) Computer software, according to claim 18, further comprising:

executable code, provided on a computer readable medium, that sets a stack pointer to an address corresponding to stack space for the particular scheduler; and

executable code, provided on a computer readable medium, that causes the processor to use the stack space at the stack pointer after executing code at the address corresponding to the program counter.

20. (Previously Presented) Computer software, according to claim 18, wherein all of the schedulers use the same stack.

21. (Cancelled)

22. (Previously Presented) Computer software, according to claim 18, wherein at least one of the schedulers is for statistical code profiling.

23. (Previously Presented) Computer software, according to claim 18, wherein a first one of the schedulers is for start up conditions and a second one of the schedulers is for steady state operation.

24. (Previously Presented) Computer software, according to claim 18, wherein executable code that causes the processor to execute code at an address sets up a return from an exception that causes the particular scheduler to execute.

25. (Previously Presented) Computer software, according to claim 18, wherein executable code that sets a program counter modifies a variable according to the particular scheduler that is chosen.

26. (Previously Presented) Computer software in combination with a computer readable medium that schedules tasks in a multitasking operating system, comprising:

executable code, provided on a computer readable medium, that chooses the particular scheduler from a plurality of schedulers, wherein at least one of the plurality of schedulers selects processes to be run from a plurality of runable processes different from the plurality of schedulers and wherein executable code that chooses the particular scheduler uses parameters that vary according to run time conditions; and

executable code, provided on a computer readable medium, that runs the particular scheduler to schedule tasks.

27. (Previously Presented) Computer software, according to claim 26, wherein executable code that chooses the particular scheduler sets up a return from an exception that causes the scheduler to execute.

28. (Previously Presented) Computer software, according to claim 26, wherein executable code that runs the particular scheduler sets a program counter to an address corresponding to code of the particular scheduler.

29. (Previously Presented) Computer software, according to claim 28, wherein setting a program counter includes modifying a variable that is modified according to the particular scheduler that is chosen.

30. (Previously Presented) Computer software, according to claim 26, further comprising:

executable code, provided on a computer readable medium, that sets a stack pointer to an address corresponding to stack space for the particular scheduler; and

executable code, provided on a computer readable medium, that causes the processor to use the stack space at the stack pointer after executing code at the address corresponding to the program counter.

31. (Previously Presented) Computer software, according to claim 26, wherein all of the schedulers use the same stack.

32. (Cancelled)

33. (Previously Presented) Computer software, according to claim 26, wherein at least one of the schedulers is for statistical code profiling.

34. (Previously Presented) Computer software, according to claim 26, wherein a first one of the schedulers is for start up conditions and a second one of the schedulers is for steady state operation.